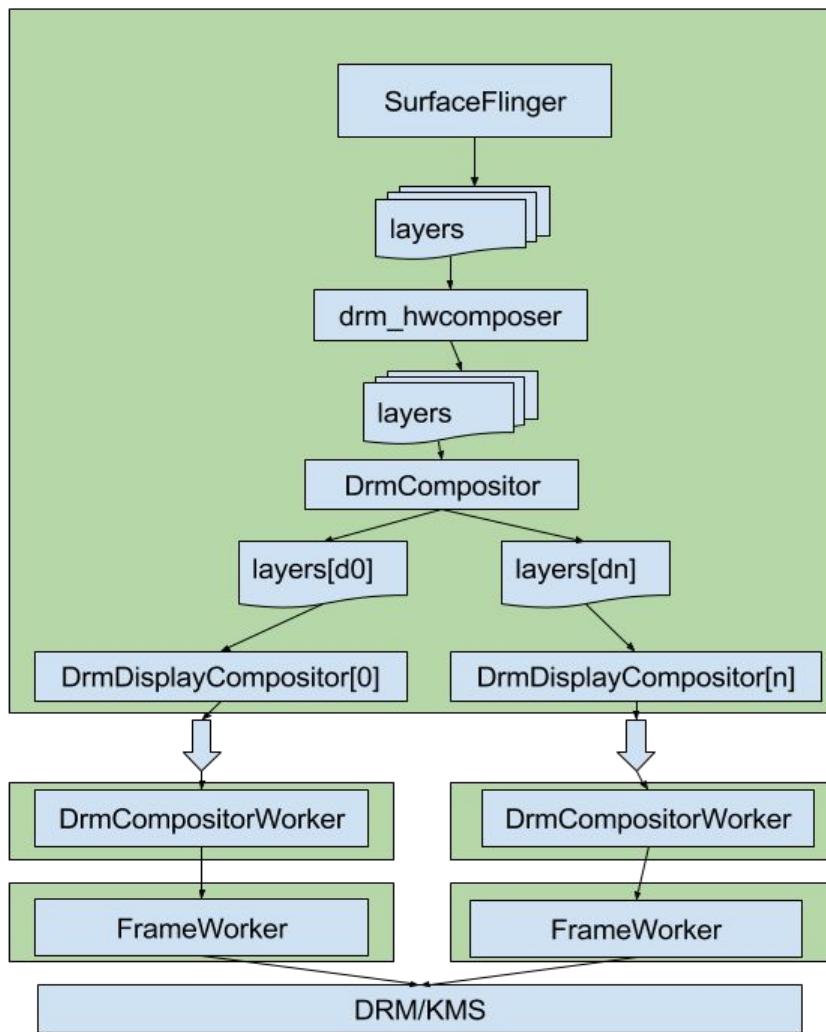


# drm\_hwcomposer

Sean Paul / Zach Reizner

# Development Timeline

- Started development January 2015
- Began with thin C implementation using legacy DRM/KMS ABI
- Converted to C++ with libdrm abstraction
- Moved to C++11 to exploit language safety features
- Added embedded GL compositor for full/partial squashing
- Shipped on Pixel C in December 2015
- Planner allows for more granular device/application specific rules
- Vulkan compositor



# Code Overview

hwc\_device\_open()

DrmResources::Init() → open(property\_get("hwc.drm.device"))  
default: /dev/dri/card0 → drmModeGetResources(fd)

DrmCompositor::Init() → Planner::CreateInstance()

for each connector...

DrmDisplayCompositor::Init() → FrameWorker::Init()  
DrmDisplayCompositor::Init() → DrmCompositorWorker::Init()

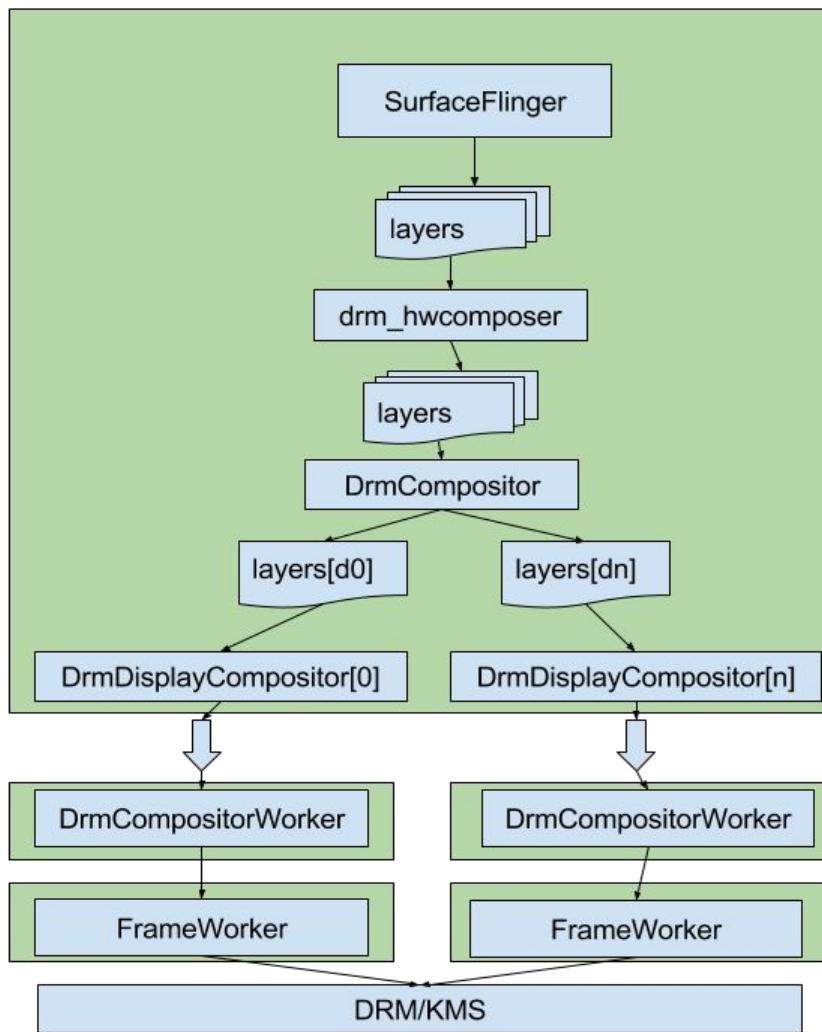
Importer::CreateInstance()

# Code Overview (continued)

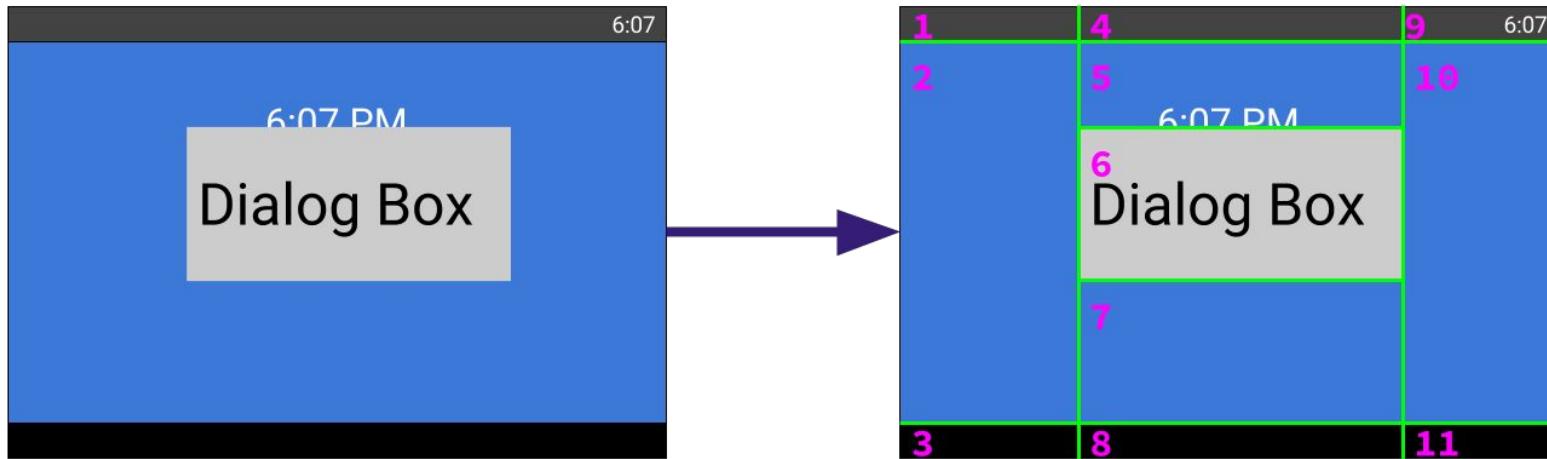
- `hwc_set(dev, contents)`
  - encapsulate everything in C+11 in case we ever fail
  - import every layer we need to composite (either with GL or overlays)
  - assign each layer a release fence
  - `DrmCompositor::CreateComposition`
  - `DrmComposition::SetLayers(contents)`
    - `DrmDisplayComposition::SetLayers(display_layers)`
  - `DrmCompositor::QueueComposition(composition)`
    - `DrmComposition::Plan`
      - for each display:
        - `DrmDisplayComposition::Plan(squash_state, primary_planes, overlay_planes)`
          - reading and writing to squash state
          - `Planner::ProvisionPlanes`
          - `DrmDisplayComposition::SeparateLayers`
          - assign fences to layers in order of completion
      - for each display:
        - `DrmDisplayCompositor::QueueComposition(display_composition)`
          - Push `display_composition` onto composition queue
  - return

# Code Overview (continued)

- DrmDisplayCompositor::Composite()
  - creates GL Compositor (called pre\_compositor in code) if needed
  - pops a DrmDisplayComposition off the queue
  - DrmDisplayCompositor::PrepareFrame(display\_composition)
    - ApplySquash(display\_comp) OR reuse the last squash
      - GLCompositor::Composite
    - ApplyPreComposite(display\_comp)
      - GLCompositor::Composite
  - queue finished frame onto the frame queue
- FrameWorker::Routine()
  - pops a finished DrmDisplayComposition of the queue
  - DrmDisplayCompositor::ApplyFrame(composition)
    - DrmDisplayCompositor::CommitFrame
      - drmModeAtomicCommit
    - Blank the display on error
    - Signal composition completion



# Rectangle Separator



```
struct DrmCompositionRegion {  
    DrmHwcRect<int> frame;  
    std::vector<size_t> source_layers;  
};
```

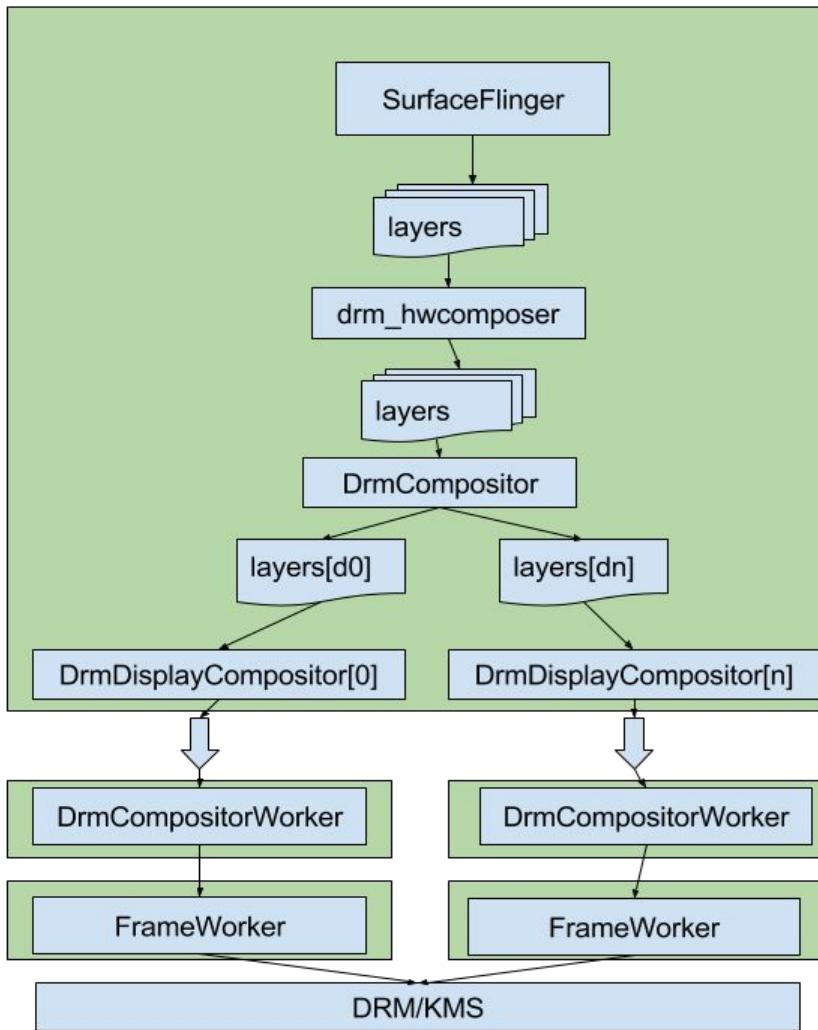
# GL Compositor

- uses separated regions directly
- generates a shader for each layer depth
- renders each rectangle region with one draw call
- no blending hardware used at all
- optimization: blending done within shader
- for layer import, uses NV hack: EGL\_NATIVE\_HANDLE\_ANDROID\_NVX
- for framebuffer import, uses standard EGL\_ANDROID\_image\_native\_buffer
- optimization: cache framebuffers using weakptr

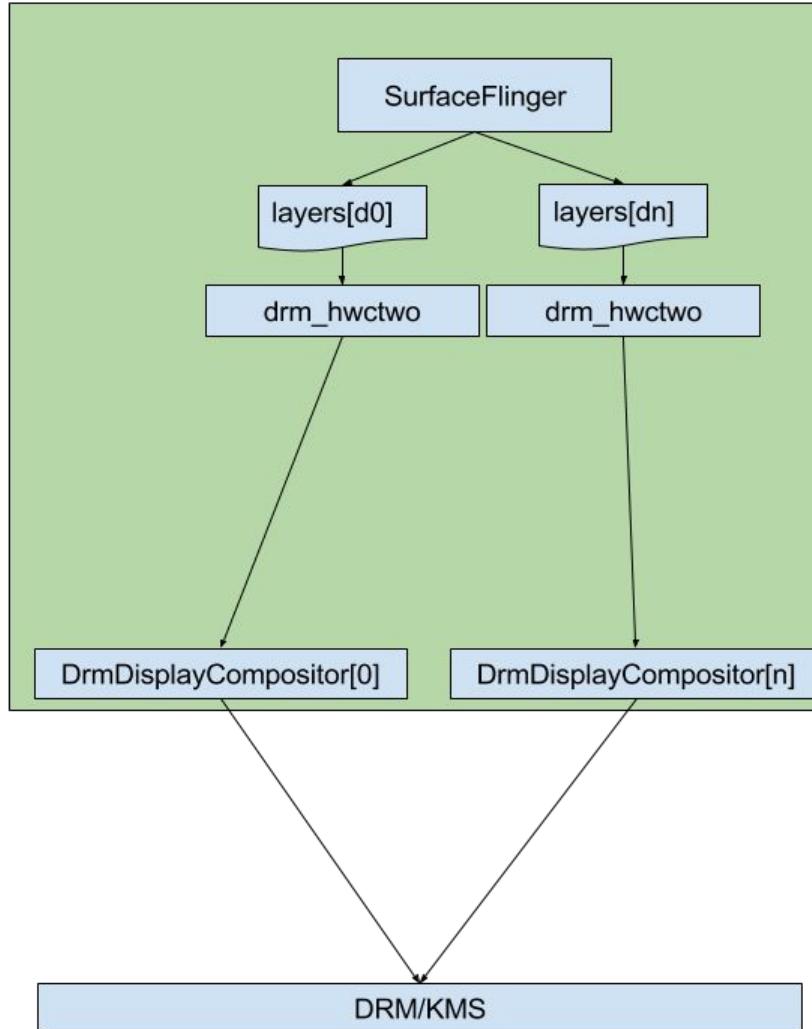
# Planner

- Introduced with Android N
- Planner runs every time the composition changes
- Platform register plan stages in priority order
- Plan stages map SurfaceFlinger layers to hardware planes
- After all stages finish, all layers should be mapped

# HWC2



# HWC2



# Contributing to drm\_hwcomposer

- Upstream source hosted on chromium.org gerrit
- External contributions welcome (thanks robher!)

[https://www.chromium.org/android/contributing-to-drm\\_hwcomposer](https://www.chromium.org/android/contributing-to-drm_hwcomposer)

# AMA